SYSTEM PARALLELISATION FOR COMPUTER VISION

Asaad A. M. AL-Sudani
Lecturer, Dept. of Elec. Eng., College of Eng.,
Univ. of Baghdad, Baghdad, Iraq.

ABSTRACT

This paper delineates the parallelisation of a computer vision system. It presents the system proposal and the relevant design phases of a laboratory - based model. This model involves special purpose hardware implementing the early stages of processing with very high data rate. It Incorporates facilities enabling the user to capture, retain, retrieve, compare, and analyse video images. The output of this hardware is to be processed by a software running in a parallel processor. The latter is a VMEbus-based multiprocessing machine accommodating the system hardware and ensures for better flexibility. It also participates in a reasonable distribution of the system processing power. The kernel philosophy here depends on the concept of modularisation to attain higher degree of design consistency. It believes that the spatiotemporal pixel variation of two adjacent video frames involves sufficient information to detect movement. This implies pixel encoding and motion parameters estimation. The system software is based on a data compressive technique (Strip Encoding of Adjacent Frames) to solve the bottlenecks problem in the whole system throughput. The research hereby attempts to attain a match in the degree of sophistication between the system hardware and software structures. This yields to make the system processing power better meets the system applications requirements. The research investigates the above presented design phases along with their logical, functional, technical, and modular specifications. The research is adequate for development in a wide range of applications (requiring parallel architectures for image processing) like: Artificial Intelligence, Features Extraction and Pattern Recognition, Expert Systems, Computer Vision and Robotic Vision, Industrial Control, and other civil and military applications.

الخلاصة

: يهدف هذا البحث الى الوضع الدقيق للخطوط العامة الخاصة بعملية تحقيق الموازاة في منظومة الابصار بالحاسوب. ينطوي البحث على مقترح لإنشاء المنظومة متضمناً أوجه التصميم ذات الصلة والخاصة بالنموذج المختبري لها. يتضمن هذا النموذج معدات خاصة تشكل المراحل الاولية من المعالجة والتي تشتمل على معدل عال جداً لمرور البيانات. تتوفر هذه المعدات على تسهيلات تمكن المستفيد من التقاط، استبقاء، استعادة، مقارنة، وتحليل الصور الفديوية. تعالج مخرجات هذه المعدات بواسطة كيان برمجي يشتغل ضمن معالج متواز. يتشكل هذا المعالج المتوازي من ماكنة متعددة المعالجات يعتمد اساسها على المسار الناقل نوع VME. تحتضن هذه الماكنة كافة معدات المنظومة بغية تأمين مرونة اعلى، وتساهم ايضاً في أقرار توزيع متوازن لقدرة المعالجة في عموم المنظومة. تعتمد الفلسفة الجوهرية لهذا البحث مفهوم التقنين وصولاً لتحقيق درجة اعلى من رصانة التصميم. تؤمن هذه الفلسفة بأن التغير الزماني-المكاني في العناصر الصورية لأثنين

متجاورين من الاطر الفديوية يحوي معلومات كافية لكشف الحركة. يتضمن ذلك تشفير العناصر الصورية وتحديد المقومات الحركية. يعتمد الكيان البرمجي للمنظومة على أسلوب يتسم بضغط البيانات (يدعى بالتشفير القطعي للأطر المتجاورة) وذلك لتجاوز مشكلة اختناقات مرور البيانات في عموم امتلائية المنظومة. يحاول البحث كذلك ان يحقق مستوى جيداً من المواءمة في درجة التعقيد بين التراكيب المعدانية والتراكيب البرامجياتية للمنظومة، وذلك لجعل قدرة المعالجة في المنظومة تفي بشكل أفضل بمتطلبات المنظومة التطبيقية. يتحرى البحث اوجه التصميم المذكورة اعلاه متناولاً كافة المواصفات المنطقية، الوظيفية، التقنية، والتراكيبية الخاصة بتلك الوجوة. يصلح هذا البحث لمديات واسعة من التطبيقات (التي تتطلب معمارية متوازية لمعالجة الصور) من مثل: الذكاء الصناعي، استخلاص المعالم وتمييز الانماط، الانظمة الخبيرة، الابصار بالحاسوب والابصار لدى الانسان الالي، السيطرة الصناعية، وما الى ذلك من تطبيقات مدنية وعسكرية مختلفة.

KEY WORDS
Parallel Processing, System Parallelisation, Software Engineering, Move Detection, Computer Vision, Machine Intelligence.

INTRODUCTION

Computer Vision is concerned with extracting Information about a scene by analysing images of that scene. It is a rapidly developing interdisciplinary field with broad and ambitious goals in many applications [Rosenfeld 1988]. It professes to be both science and engineering. As a science, it seeks to articulate the principles and algorithms by which visual Images can be analysed for a wide spectrum of useful information. As engineering, it explores the design and validation of complete systems that efficiently process highly complex spatially distributed signals into a simpler and stabler set of symbols. In its broadest definition, it involves everything that helps in making sense of the two-dimensional (2D) data[Li and Kender 1988]. Although it is fairly easy and inexpensive to assemble a computer vision system, it has proved surprisingly difficult to achieve a vision capability in machines, even to a limited degree. This is not to imply that the scientists are not using all sorts of vision systems and motion detectors in a variety of applications. Yet, the ability to discern objects, ascertain their motion, and navigate in the three-dimensional (3D) space through the use of machine vision involves an appreciable degree of sophistication [Aggarwal and Nandhkumar 1988]. Incorporating such vision in a machine is by no means a straightforward task given the widespread availability of microcomputers, digitising cards, and solid-state cameras [AL-Sudani 1998i].

To avoid confusion and ambiguity, a matter of interest here is to highlight the organic relevance between the two significant and closely related terminologies; the "Computer Vision" and the "Digital Image Processing". The term Computer Vision is young, arose in the early eighties, challenging the preceding (older) term Digital Image Processing, and holding its first international conference in 1987 [Li and Kender 1988]. Image Processing is generally concerned with the manipulation of images by a computer. The Digital Image Processing may be defined as the act of subjecting numerical representation of graphic objects to a series of operations in order to obtain a desired result. For operational convenience this generally incorporates computers of "digital" type. Image Processing is hence used to be called as Digital Image Processing [AL-Sudani 1998i].

Despite the fact that the two terms are both in current common use, yet the more comprehensive term Computer Vision became for the time being more dominant. It is defined here as the processes

that are concerned with developing pictorial/nonpictorial Information about natural/artificial scene by analyzing and/or processing images of that scene. From such definition, and as the technology progressively advances, the distinction between the two terms becomes smaller and smaller. In this extended sense, the Computer Vision topic may hereby be considered to involve several classes of the "Parallel Digital Image Processing". In most references, the three categories; Generative Graphics, Image Processing and Cognitive Graphics (Scene Analysis) form altogether the field of "Computer Graphics"[AL-Sudani 1998i]. As sequential (serial) uniprocessing machines can not achieve the speed required by most image processing applications, parallel architectures accommodating the relevant parallel algorithms have therefore to be utilised [AL-Sudani 1998ii][Stout 1988]. This orientation is sometimes denoted as Parallel Digital Image Processing.

Broadly speaking, Computer Vision Involves two categories of image processing; Low-Level Image Processing, and High-Level Image Processing [Faugeras 1983][Lee 1988]. The first category, Low-Level Image Processing (also called Early Vision) is concerned with decoding 2D Images to obtain specifications of the 3D surfaces. In other words, it is concerned with extracting 3D information from 2D Images. This comprehends depth, surface orientation, distance between the surfaces and the viewer, texture, reflectance, and parameters of the object motion. Unlike the other category: it involves bottom-up processes, its processes are independent of each other, it uses distinct hardware pieces, and it is considered to have ill-posed (ill-conditioned) problems [Bertero et al. 1988] (the solution of the ill-posed problems often does not exist, or exists multiply, or varies discontinuously with changes in the data).

This category of processing may in turn be subdivided into two classes of low-level image processing; Image Pre-processing, and Image Understanding [Lawton and McConnell 1988] [Weems 1988].

Image Pre-processing involves operations (algorithms) that are to be applied on an image or on segment(s) of that image. This may be represented in three types of operations: (i) Pointwise Operations, (ii) Neighborhood operations, and (iii) Global Operations. The pointwise operations are exemplified in Histogram Equalisation for visual surface interpolation, and in Thresholding for obtaining binary images. The neighbourhood operations are concentrated on: Thining, Averaging (Smoothing by convolution template) for noise elimination, and Edge Detection/Enhancement for shape recovery from shading. The global operations incorporate:- Image Rotation, Scaling, and Filtering by Fourier Transform [Agrawal and Jain 1981][Aloimonos 1988][Bajcsy 1988].

Image Understanding (IU) (also called as Scene Analysis [Rosenfeld 1988] or Mid-Level Computer Vision [Aloimonos 1988]) comprises relatively higher level operations for different tasks. This may involve: Features Extraction (Structure from Stereo and Structure from Texture), Recovery of Motion (Structure from Motion), Computation of Optical Flow, Pattern Recognition, etc.[Faugeras 1983] [Weems 1988].

The second category (High Level Image Processing) is usually concerned with determining the "meaning" of the scene. It is represented in the Machine (Artificial) Intelligence, Expert Systems, and Knowledge-based Systems. Examples of applications in this category are the Navigation in the environment, Manipulation of objects, Object recognition, as well as Reasoning (Inferring) about objects[Rosenfeld 1988][Faugeras 1983].

A task that is closely related to the issue is the "frame grabbing". In this context, pictures can be picked as analog images via video camera (Videcon). In order to process a video image on a digital computer, the task requires to convert that analog image into a digital image (called digitised image or pictorial data). This image capture is usually achieved by an image digitiser (A/D frame grabber), and saved on an image buffer or disc (frame store). The process incorporates obtaining a discrete 2D array of numbers representing light intensity (brightness or colour values) corresponding to the discrete grid of points in the image plane. Or, more precisely, corresponding to the average values of brightness of the neighborhoods of those points. The elements of the array are usually called

"pixels" (PIXELS is a short word slightly abridged from "PICtureS ELementS"), and the brightness value is called the "grey level" of the pixel.
In applications requiring colour information, specific colours values are measured at each pixel to represent the composite brightness level at that pixel. This means that each colour pixel has a brightness level that incorporates a set of "n" spectral bands, or in other words each colour pixel has n-tuple levels of the specific colours values [Rosenfeld 1988].
Within the above context, the main objective of this research is to investigate the optimal parallel architecture with its associate Inherent parallelism to establish a computer vision system. This involves the structured design of such vision system plus investigating its development to incorporate an extendable library of utilities.

ORIENTATIONS IN COMPUTER VISION FIELD

There are innumerable ways by which to see, and at least as many means by which to build a seeing machine [Li and Kender 1988]. Computer Vision attempts to capture for its own ends, in math and in silicon, all such useful ways and means of sight. Like other highly experimental emergent disciplines, computer vision borrows freely from the theories and practices of its many resources [Faugeras 1983]. Spectral, spatial, and temporal resolutions span enormous ranges. From biology and psychology, particularly psychophysics, it derives insight into which basic computational visual processes have proven most valuable over the long field test of human history. From mathematics it adopts subfields both continuous and discrete. From numerical analysis, the calculus of variations, and solid geometry, among others, it finds ways of inferring the literally deep structure of objects from the flatness of the image. From graph theory, statistical Inference, and pattern recognition, among others, it constructs algorithms that match the always inexact input to the most likely construct of previously known parts, relations, and purposes. From electrical engineering, it takes signal processing for its front ends, VLSI (and system design) for its special purpose highly parallel hardware, and control theory for its robotic real-time systems. And from computer science and artificial Intelligence it adopts data structures, search techniques, and complexity analyses, as well as the software engineering tools to enable its theories of sight to be quantifiably tested and efficiently explored [Li and Kender 1988].
Computer vision approaches generally fall into four groups. The first approach deals with establishing the vocabulary, concerns, methods, and devices of the field. The second approach discusses the computational and theoretic foundations for various stages of image understanding, from early vision to geometric classification. This group, rather mathematical in flavour, places particular emphasis on the efficient and accurate recovery of the third dimension by using various model-based assumptions about the external environment. The third approach discusses parallel hardware architectures that exploit the natural 2D structure of many image processing algorithms. The final (fourth) approach describes computer vision systems at work, with the difficulties they face before and during operation ; especially in the software engineering side provoked by such systems and environments.
This research is deemed to belong to the third category of the above mentioned approaches.

SYSTEM DESIGN CONSIDERATIONS

To address the various issues of the computer vision system integration; how to make use of proper type among several types of data structures; how to organise the system via active and intelligent sensing; and how to support it with robust architectures and equipments, several design considerations have here been taken in account [Rosenfeld 1988][Faugeras 1983] . This is discussed briefly as follows:

1- The input images in a computer vision system are usually numerical-valued pixel arrays, with pixel values as grey levels. Many other types of arrays, derived from the input images, may also be used in vision systems. These include numerical (or even vector-valued) arrays such as:

filtered images, transforms, intrinsic images, and Gaussian images. This may also involve "symbolic images" in which the pixel values are labels rather than numerical quantities (e.g. thresholding binary images, feature maps, arrays resulting from connected component labeling, etc.). In this concern, regions or features that have been extracted from an image in the course of analysing it can also be represented by binary images. Yet they can alternatively be represented in more compact ways such as by sets of coordinates, by run length codes, by boundary/chain codes, by medial axis transformations, or by quadtrees. Similarly the 3D objects that are to be recognized in an image can also be represented using various types of geometric data structures, including surface models and solid models. On the other hand, collections of image parts (or object parts), their properties, and the relations among them can be represented by labeled graph structures. In such kind of representation, the nodes represent the parts, the node labels represent property values, and the arc labels represent relation values [Rosenfeld 1988].

2- Each of the above mentioned data types may need many different kinds of operations performed on it in a general vision system [Stout 1988]. On array data, in particular, the following types of operations are usually performed:

a- Point and local operations (e.g. grey scale transformations, convolutions, morphological operations, etc.).
b- Statistical computations (e.g. histogramming, texture analysis, etc.).
c- Transforms (Fourier, Hough, etc.).
d- Geometric operations (e.g. perspective projection, warping, etc.).
e- Extraction of geometric entities (e.g. boundary curves or regions such as boundary following, region growing, etc.).

It is less well understood what types of operations may need to be applied at the level of geometric representations, but an initial list might include the following:

- Boolean combinations (unions, intersections, differences).
- Derived sets (convex hulls, Voronoi diagrams, and skeletons, etc.).
- Visibility and mobility computations (path planning, etc.).
- Geometric property computations (connectivity, moments, etc.).

The operations that might be needed at the graph level are quite general. They comprehend graph search and path finding, clique finding, sub-graph isomorphism testing, and others.

3- Many kinds of parallel architectures have generally been used for computer vision objectives[Al-Sudani1998i][Al-Sudani1998ii][Maresca et al. 1988][Burt 1988] [Bajcsy 1988][Dew et al. 1988][Hwang and Briggs 1985][Lin and Kumar 1990][Hayat et al. 1992][Morris et al. 1989]. From topological, structural, operational, and functional view points this (currently) includes the following:

a- Pipelined systems like the various types of the 'cytocomputers' and the 'systolic array processors'. Such systems perform sequences of operations on a stream of input data. As soon as one operation has been performed on a given part of the data, the next operation can be initiated on that part. They are especially useful in morphological image processing, where long sequences of local operations are to be performed on a given image [Dew et al. 1988].
b- Mesh-connected systems like the Cellular Logic Image Processor (CLIP), the Massively Parallel Processor (MPP), and the Geometric Arithmetic Parallel Processor (GAPP). Such systems can perform local operations on an entire image simultaneously [Hwang and Briggs 1985].
c- Trees and pyramids like the Non-Von, the Ultracomputer, PAPIA, and numerous others [Hwang and Briggs 1985]. These systems can perform global operations (such as statistical computations, or large-kernel local operations) using divide-and-conquer techniques. This is done in number of processing steps that grow only logarithmically with the image or kernel size.

d- Hypercubes like the N-cube and the Connection Machine [Maresca et al. 1988] [Hwang and Briggs 1985]. Such systems combine the advantages of meshes and trees (or pyramids), and can also perform other types of global operations (such as transforms) in an also logarithmic number of steps.

e- Shared-bus / shared-memory (multiprocessing) machines like the CYBA-M, Cmmp, and the Butterfly. By using a suitable interconnection network, such systems can simulate architectures in which the connections are hardwired to include the above four types of architectures [Maresca et al. 1987][AL-Sudani1988].

SYSTEM MODEL

Within the confines of the above considerations, and to meet the objectives stated in the first section, **Fig.(1)** presents the abstract model of the Compute Vision System. With some elaboration, **Fig.(2)** delineates the processes of the system model. This incorporates: Image Capturing and Frame Grabbing, Image Buffering and Frame Storing, User-SystemInterfacing, Image Transformations, Pixel Differencing and Strip Encoding (strip Encoding is a developed technique for features extraction that ensures for data compression), Object Encoding, Motion

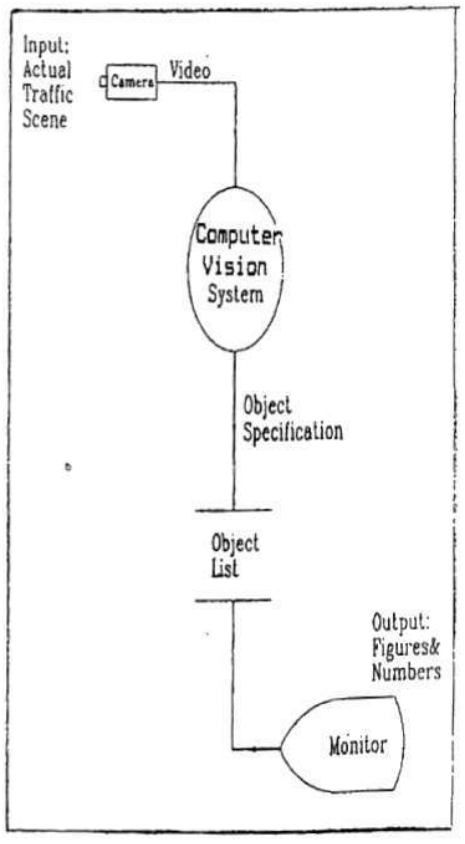

Fig.(1) Abstract Model

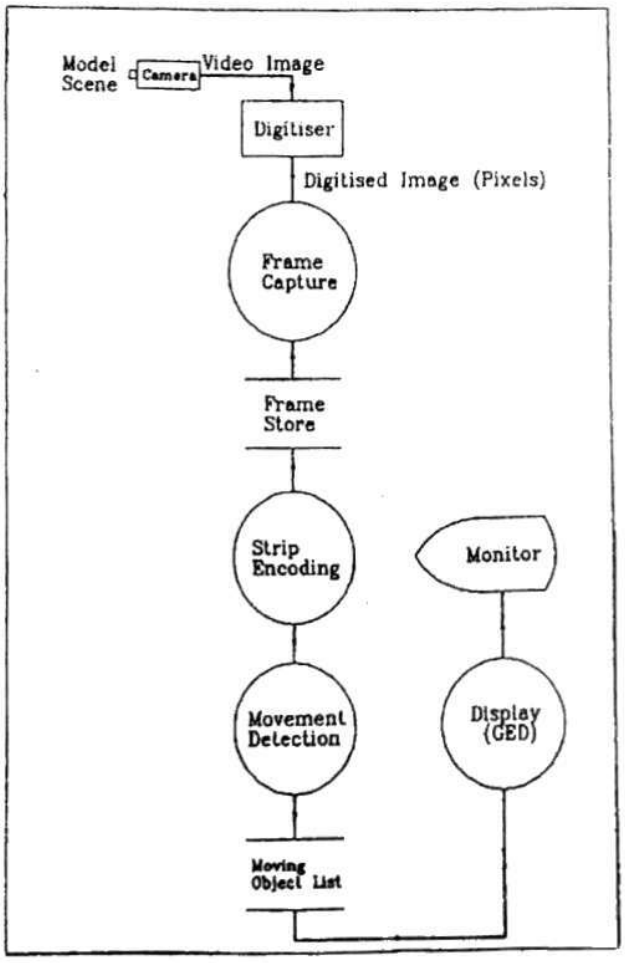

Fig.(2) System Model

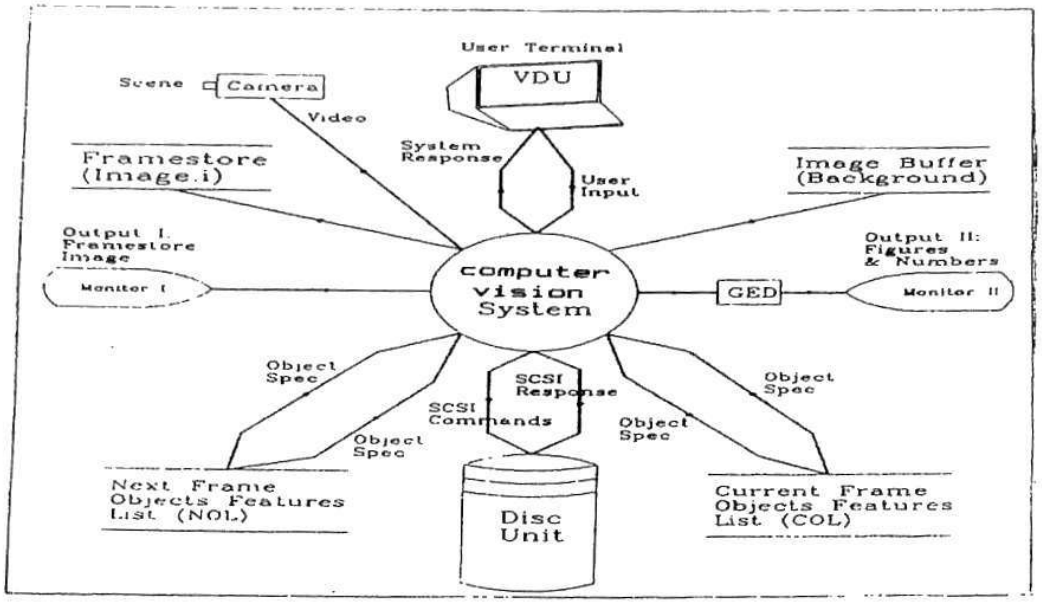

Fig.(3) System Context Diagram
(GED = Graphics Editor)

Detecting, and Output Displaying. On the other hand, **Fig.(3)** illustrates the system's context diagram. It demonstrates the outside world enclosing this system and gives a flavour about the nature of that world.

To realise the research's system model, three phases have to be considered; System Parallelisation, Hardware Configuration, and Software Structured Design. This is presented in the following sections.

SYSTEM PARALLELISATION

It has been denoted earlier that conventional uniprocessing machines are not able to support processing applications with very high data rate. This follows to resort to parallel processing architectures and to exploit parallelism inherent in the image processing operations. As articulated in article 3 of third section here, but specifically from an architectural view point, parallel computing structures involve several 'classes' like: SIMD Array Processing, MISD Pipelining, MIMD Multiprocessing, VLSI Algorithmic Processing, and Data Flow Computing [Dew et al. 1988][Hwang and Briggs 1985]. On the other hand, parallelism inherent in image processing operations (motivated in the first section and article 2 of the third section) also comprehends several levels (types) such as: Pixel-Bit Parallelism, Neighbourhood Parallelism, Image Parallelism, and Operator (Task) Parallelism [Stout 1988]. For operational convenience, a practical correspondence has been noticed in some of those architectures to exclusively support some of these parallelism types. This has been highlighted in the following paragraphs.

The Pixel-Bit Parallelism arises immediately in the traditional computer technology. Here, all the bits of the grey level pixel are fetched in parallel by the acting processor. So it is inherent in all computers except the 'bit-serial' machines such as the DAP and CLIP series array processors.

The Neighbourhood Parallelism means that a processor can simultaneously look at not just the immediate pixel but also its neighbours, without incurring any penalty for storage look-up. This requires immediate parallel access to at least one bit-plane of the neighbourhood. The process computes one output pixel and then proceeds to the next neighbourhood. As evident from its

definition, such parallelism can be better met through special hardware facilities as in the existing MISD pipeline machies like the Cytocomputer.

Image Parallelism means the provision of many processors over the whole image. Such process may be performed either synchronously as in the GRID and CLIP array processors, or asynchronously like in the PICAP 11 and the VME bus-based multiprocessors. The processor here may vary from a general purpose CPU to extremely simple bit-serial type. In this scheme, pixels are allocated to a processor either statically (before run time) or dynamically (by controlling processor).

The Operator (Task) Parallelism is quite adequate with the MISD pipelining class, where each data set (as motivated in article 3a of the third section) is passed from one set of processors to another. In fact the processing task here is broken down into a number of subtasks, each to be performed by a different processor working altogether in parallel. Here each stage requires a limited amount of local memory. The first processor receives data fed to the pipeline (buffered in its local memory) and computes the output data to be buffered into the next stage local memory. Whilst this stage is busy in processing data fetched from its memory, the first stage is in turn busy in processing the next amount of data coming to the pipeline. The same action is performed in cascade between the second and third stages under control organised by the system clock. The cascade process continues till finishing the whole data fed to the system pipeline. The particular advantage of pipelining class is that each processor can be tailored for one processing task and does not need to be equally good in all situations that may arise. Also, the pipeline processors need not to be identical. Examples on such systems are the previously mentioned Cytocomputers and the AP-120B pipeline machine.

The VLSI algorithmic processors have been devised by the rapid advent of the VLSI microelectronics technology. It represents a new architectural horizon in implementing parallel algorithms as a direct hardware (VLSI computing structures). The new high resolution lithographic technique has made possible the fabrication of more than 1 million transistors in an NMOS chip. Most proposed VLSI arithmetic devices are for vector and matrix type computations. Potential applications of the VLSI computing structures are recommended for real-time image processing. This includes globally structured arrays and modular computing networks. The use of the VLSI technology in designing high performance multiprocessors, pipelined computers, systolic arrays, and massively parallel processors is currently under intensive investigation in both industrial and research establishments. Examples here (as mentioned in article 3b of the third section) are the MPP array processor and the GAPP pipeline systolic array (single CMOS chip).

All the above exemplified computing structures are classified within the conventional Von Neumann's concept of the 'Control Flow' machines. Whereby, the program instructions are executed sequentially and their order is governed by the instruction pointer (program counter). This process is inherently slow. To exploit maximal parallelism in the program execution, 'Data Flow (Data Driven)' machines were devised. The basic concept here is to enable the execution of an instruction whenever its operands become available. This means that the instructions in the program are not ordered; the execution follows the data dependency constraints. Theoretically, maximal cocurrency can be exploited in such machines; costrained only by the availability of the relevant hardware resources [Hwang and Briggs 1985].

From the above discussion, it might be clear that an optimal vision system has to involve a combination of some various types and architectures for better realisation of parallelism. However, for practical reasons, it is sometimes necessary to deal with the issue in a cost-effective manner. This means to customize hardware and software solutions with the ultimate possible flexibility, and to take in consideration the effective practical constraints. Hereby, the envisaged organisation to achieve parallelism in the system design of this research requires a reasonable amount of storage in each processor and an efficient communication network between processors. Ideally, each processor should have enough storage to allow a copy of entire image to be stored and have a direct access to other processors memories. Nevertheless, the complexity of the communication network grows rapidly as the number of processors employed in the system increases [Morris et al. 1987][AL-

Sudani 1988]. This may cause degrading the overall performance when performing operations that involve inter-processor communications between remote processors. In practice, thereby, a suitable compromise between these basic architectural features must be adopted by understanding the nature of the computer vision algorithms. Furthermore, the better design here is to fully exploit parallelism, both in software and hardware, in the interest of maximising the computational throughput. In general, experience and knowledge gained through research work in this field have led to design special purpose systems by recognising the nature of parallelism available in image processing algorithms. Thus, by looking at the nature of the existing architectures, one can generalise the aspects of the vision system they support.

SIMD array processors only employ communication between neighbourhood processors reducing the complexity of the communication and control of the system at the expense of the degraded flexibility. In contrast, MIMD multi-processors employ fast communication networks between processors which effectively increases the flexibility at an expense of increased complexity in control and communication, which rapidly grows with the number of processors. As a result, SIMD array processors employ the order of hundreds or thousands (or even tens of thousands) of processors as compared with tens or hundreds of processors in the case of MIMD multiprocessor machines.

Although SIMD array processors and MIMD multiprocessors are very powerful, yet some critics believe that their hardware parallelism doesn't match to the current imaging technology. Since the data from conventional image sensors come in serially, the start of processing must be delayed at least one video frame. Herewith, the concept of the MISD pipeline architecture has been deemed to match the vision system with serial data coming from image sensors [AL-Sudani 1998i]. Albeit, MISD pipeline architectures tend to use processors specialised for particular type of problems. Generally, SIMD array processors are quite special purpose while MIMD multiprocessors are often general purpose. The MISD pipelines fall in between; they might involve processors of the dedicated-different specifications general purpose type.

In the MISD pipelines, subtasks have to be allocated to the successive processors, and they have to be performed in a single data stream. Unequal execution times of subtasks will create delays in each processing stage degrading the overall performance. Another problem which may arise in MISD systems would be the insufficient cocurrency in the algorithm to allocate subtasks to all the available processors. An MISD pipeline with a huge number of processors will have to face this problem [Hwang and Briggs 1985].

On the other hand, systolic arrays which operate in a pipeline fashion, are optimised in their design (in the number of processors and in the communication network) to exactly fit the available parallelism of a particular task. Such a development of systolic architectures has led to various special purpose systems capable of implementing various fixed and well understood computational routines .

The crucial feature of the SIMD array processor is its algorithm-structured architecture, which allows efficient processing for low level operations [Lin et al. 1990]. For such operations (like convolution and edge detection) it is possible to configure highly optimised SIMD architectures down even to pixel level. Higher levels which process lines, shapes, and 3D models, involve however totally different data structures with very little amount of parallelism and may poorly match SIMD architecture. In contrast with that, and due to flexibility and possibility of using task parallelism at high level operations, MIMD multiprocessing architectures can be employed using multiple copies of the image or one or more shared copies. For instance, in case where the aim is to locate two or more distinct objects in an image, each object can be assigned to a processor or a set of processors to search for it[Morris et al. 1989].

Another disadvantage in the SIMD array processors is the degradation of performance if branching operations were to be performed. When a branch point occurs, several processing elements will be in one state and the remainder will be in another. The master controller can essentially control only

one of the two states; other goes idle meanwhile. This is not the case in MIMD multiprocessing systems, since each processor has its own control unit and hence avoids such kind of degradation. A further point of interest is that even in low level image processing, some algorithms are more suited to fit the SIMD array processing and some to the MIMD multiprocessing [Lee 1988][Bertero 1988]. As mentioned earlier, furthermore, processing units used in the MIMD multiprocessors are normally general purpose microprocessors. Hence, the system designer wishing to use them in an architecture won't have to go through the design cycle of their fabrication. For economical reasons, moreover, modern semiconductors technologies (fast, dense, etc.) have usually been recommended in their fabrication. Yet this is neither the case in the SIMD array processors or the MISD pipelines, nor the case in the VLSI algorithmic processors or the Data Flow computers. Processing elements in those classes are of dedicated/special purpose type, and their fabrication phase must be taken in account if the designer wishes to use them. In fact, this fabrication phase is so expensive and its logical validation costs quite a lot, especially for the VLSI algorithmic processors and the Data Flow computers. The implication here is that researchers working with those three classes have to face difficulties in upgrading modern additional technologies.

From the above stated elaboration, the conclusion here is that the class of MIMD multiprocessing architectures provides a powerful and cost-effective approach to realise parallelism in the computer vision system presented in this research. Furthermore, the higher degree of flexibility available in the MIMD multiprocessors makes those systems expedient for both low level and high level image processing. On the other hand, moreover, the nature of operation of the MIMD machine is quite fit to suite the mode of frames processing exhibited in the system model of **Fig.(2)** and the context diagram of **Fig.(3)**.

Thereby, a VME bus-based MIMD multiprocessing machine [VMEbus 1985][SBC1 1986]. would be quite adequate to provide for the hardware requirements accommodating the Computer Vision System. Although it may seem to face the problem of bottlenecks due to the high data traffic, yet each processor with its own local memory will avoid the excessive need for data communication between processors, and hence resolves the problem of contention. This approach will lead to a simple 'loosely coupled' system since communication between remote processors will be very rare or even not necessary at all in some image processing operations.

The last point of consideration here is the "type of parallelism" most suited with the chosen architecture. Recalling the classifications at the beginning of this section, the type Image/Image Segment(s) Parallelism will be chosen here as an associate type of parallelism to exploit optimal concurrency. This provides for system parallelisation emphasised in this research.

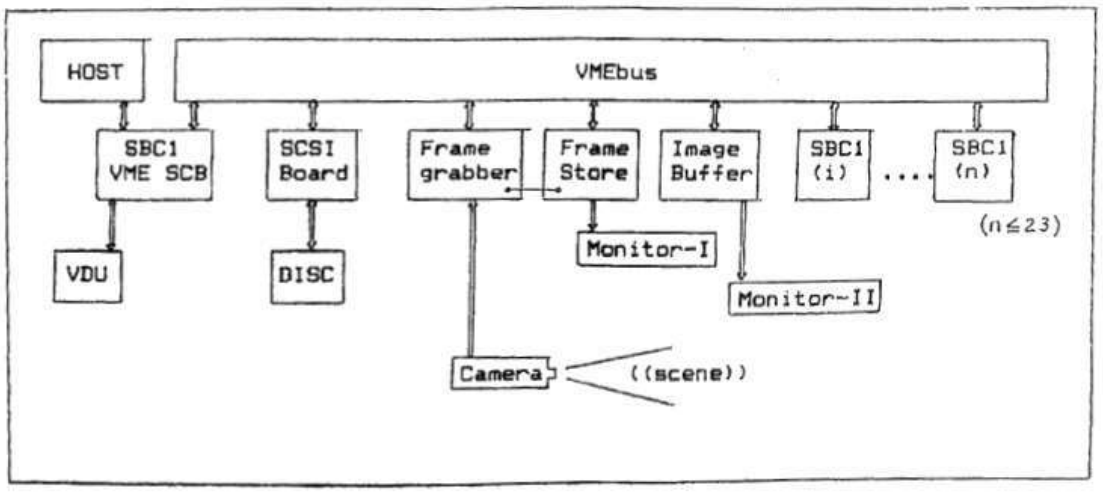

Fig.(4) System Hardware Configuration

SYSTEM HARDWARE

Images usually contain vast amounts of information; enough to challenge the capabilities of the most powerful computers. Yet only a small fraction of this information may be pertinent to a given vision task. It is thus expedient, often essential, that a vision system avoids or discards excessive details at the earliest stages of analysis. This process is sometimes referred to as 'data compression'. It is also equally important that analysis be performed with algorithms which are structured to be highly efficient, and which can be performed effectively on the system's computing hardware. Within limited computing resources, the system designer may justifiably ask: how should a system be organised and controlled to make most effective use of those resources? This paper attempts to answer such a question through presenting a computer vision system that makes careful use of selective analysis and fast algorithms to achieve higher efficiency.

In due course, **Fig. (4)** illustrates a VMEbus-based MIMD multiprocessing machine that is so convenient to support the hardware configuration requirements of the computer vision system of this research. A VDU dump terminal is hereby connected to one of the two RS-232 serial I/O ports of the Single Board Computer (SBC1). So the operator will be able to use it to call the system software stored in the host to run the system. According to the VMEbus specifications [VMEbus 1985][SBC1 1986], this SBC1 will act here as the system controller board (SCB) as it is plugged in slot 1 of the bus backplane (motherboard); the on-board arbiter jumpers have thus to be accordingly activated. To arrange for the requirement of parallelism, some process-dedicated SBCIs have to be plugged in the VMEbus backplane. The VMEbus here provides for an efficient interconnection system with high performance. To arrange for system imagery, one may use a proper video camera (videcon), frame grabber (image digitiser), mointor(s), 0.5 MB framestore, 2MB RAM, in addition to 10 MB disc with the associate SCSI board. The camera of 768×575 pixel store and 4:3 aspect ratio connected to the frame grabber can capture (in response to instructions implied in the system software) images and directly send them to the frame store (FS) through the coupling in order to be directly displayed on the monitor. The frame grabber board involves control/status registers through which the software can communicate (read/write) to this board. The 2MB RAM board represents here the required system image buffer (IB). An additional 2 MB RAM may be plugged in the VMEbus cage to provide the system with the required buffer store (BS). The system image processing transformations can be performed (as appropriate) using both of the global IB 2 MB RAM and the local SBC1 0.5 MB RAM. The Small Computer System Interface (SCSI) board will be here the necessary disc controller between the 10 MB Winchester Removable Hard Disc and the VMEbus. The captured images can be stored in the file store provided by that disc. As the system software is written in high level language, the host computer can be accessed through the second I/O port of the SBC1 card. This will provide the system software with the compilation requirements necessary to run the system. After compilation, the system software can then be down line loaded from the host into the SCB 0.5 MB local RAM. The system can thus be disconnected from the host, and the operator will be able to run the system activities through the dump terminal.

SYSTEM SOFTWARE

A primary concern of the software design is to achieve high utilisation of parallelism whilst keeping the load on common memories within allowable bounds. The use of common memories is therefore critical and will serve for several functions. Amongst these are:

- Task allocation to processing elements.
- Command request to the host.
- Backup memory for paged data and downline loaded code, and
- Interprocess communication.

On the other hand, and for pragmatic reasons, a phased approach can be adopted to the software design to exploit parallelism through extensions to conventional (procedural) languages. Although

this might be regarded as a rather 'conservative' approach, yet some application experts see virtue in this as it is hereby possible to exploit parallelism without needing to learn new languages. Based on such policy and within the confines of the system model proposed in **Fig.(2)** (and environmented in the context diagram of **Fig.(3)**), the system software in this research involves the five following modules.

1- The Interactive Module: provides a user interface by passing the choices menu supplied by the Control Module to the user. It contains procedures to deal with choice handling and parameters acquisition. This incorporates displaying menus and messages, as well as reading/writing integer, real, and string characters.
2- The Disc Manager; handles the image file store. It involves procedures to initialise the SCSI board, get the disc status, and select a required peripheral. It also involves procedures to read, write, and delete images from the disc.
3- The Features Extraction Module: deals with features extraction task. It incorporates procedures for image reading/writing, image filtering (averaging for noise reduction), as well as edge enhancing. It also involves pixel differencing (PD) and strip encoding (SE), in addition to forming the current and next frame objects features lists (COL and NOL).
4- The Movement Detection Module: achieves the task of move detection. This comprises a procedure to get a copy of each of the objects features lists (COL and NOL), and to compute objects velocities through comparing the two lists.
5- The Control Module; contains a procedure to handle menus displaying with the corresponding choices, obtaining when necessary information from other modules. It arranges for features extraction and movement detection. It also requests image capture and organises image preservation where enough memory is available.

In the simulation of the system model processes (as motivated earlier), the relevant software procedures can be preliminarily coded in a common high level language (eg. Pascal, C, etc.)with excursions into machine code where demanded by efficiency constraints or to enable access to specific store locations for driving the hardware. A second phase of simulation may involve coding in parallel processing languages (eg. Glypnir, Vectran, Tranqual, etc.). This approach has been found very useful for testing individually a single aspect of an algorithm; especially when the scene is being set to be sensitive to a specified step in that algorithm.

SUMMARY AND CONCLUSIONS

The primary objective of this research is the investigation of knowledge that will lead to more effective realisation of computer vision system. This requires better incorporation of parallel architectures associated with better utilisation of inherent parallelism. Planning and progress reviews form a major part of the further works. The research plan involves the following:

- Identifying the research objectives.
- Investigating the possible alternatives for system parallelisation.
- Developing the specifications of the system architecture to present the system skeleton.
- Establishing knowledge about the system through:

a- Building the system model prototype to show the system nature and behaviour.
b- Investigating some relevant works for performance assessment.
c- Applying specific solutions for problems arising throughout the system hardware and software design phases.

The following paragraphs are intended to reveal several sides of the parallelisation applied on the system model. These sides have been remarked during the research different stages. They may give a flavour about the research's merits/demerits concerning the system parallelism, system architecture, and the system hardware and software. They also articulate some relevant future works.

1- It is well understood how to use most of the architectures presented in section V to speed up processing at the array level. Yet much less is known (unfortunately) about how to use them to achieve speedups at the levels of geometric representations or graph structures. Whereby, it is commonly believed that the speedup of array level processing is all what is needed, since that level involves the largest amounts of data (images, transformed images, etc.), where as the remaining levels involve only small numbers of entities (image parts, etc.). This belief may be dangerously misleading. Processing of image parts, for instance, may lead to 'combinatorial explosions' of derived parts, and of relations among collections of parts. So it should not be assumed that the computational bottleneck in vision is entirely, or even primarily, at the array level. It is thus important to learn how to use the proposed architectures at higher levels, or falling this to design new architectures appropriate for those levels.

2- The computer vision system presented in this research can be described as consisting of stages like; features extraction, recovery, segmentation, etc.. And obviously, various 'techniques' are possible to be applied at each of those stages. Ordinarily, in a given system, only one or perhaps at most few techniques can be used at a given stage. If computational resources permitted, it would be better to use 'multiple' techniques and to combine the results into some form of consensus. This is an example of the general problem of combining evidence obtained from multiple resources, which is an active research topic. Most vision systems use images obtained by only one type of sensors, multisensor systems are thus also a subject of active research. In fact treating the vision process as a 'fixed' sequence of stages is itself a simplification. Ideally, the stages should be closely integrated; the results obtained at a given stage should provide 'feedback' to modify the techniques used at previous stages. This is rarely done in existing vision systems, and as a result, little is known about how to design systems that incorporate feedback between stages.

3- Parallel computing structures vary in several notions like architecture, computation, cost, and ease in programming. They might be described as follows.

4- Pipelined processors are dominating the commercial market in both business and scientific applications. Pipelined computers cost less and their operating systems are well developed to attain better resources utilisation and higher performance.

5- Array processors are mostly custom designed. For specific applications they are so effective. The performance/cost ratio of such special purpose machines may be low. Programming on array processor is much more difficult due to the rigid architecture.

6- Data Flow computers and VLSI algorithmic processors are computing structures that have their own specific constraints; they are still under development.

7- Multiprocessor systems are more flexible for general purpose applications. Pipelined multiprocessor systems represent the state-of-the-art design in parallel processing computers. Many of the computer manufacturers are taking this route in upgrading their existing systems [Hayat et al. 1992].

8- A homogeneous system is easier to program and eliminates the 'connector' problem that arises when getting two dissimilar processors to effectively communicate. The symmetric system usually can better facilitate error recovery in case of failure.

9- Any bus system has a limit on its capacity (throughput), that basically depends on the three main bus factors; bus width, bus protocol, and bus speed. The first and second factors are related to the bus specific design. Whereas the third is relevant to both of the other factors set in the bus design. It can thus be controlled and henceforth improved through using specific techniques such as 'pipelined addressing' [AL-Sudani 1988]. Here by, the speed with which slaves respond to an access can sometimes be improved by providing the slave with the address of the location to be accessed earlier in the cycle. VMEbus provides for this by enabling the address to be pipelined [VMEbus 1985]. In this technique, the address of the next location to be

accessed is transmitted on the address bus whilst data from the previous access is still available on the bus.

10- An MIMD multiprocessing system (with a higher degree of parallelism) incorporating an efficient communication network (with minimal bottlenecks) will be an appropriate solution for real/non-real time computer vision. Hereby, the experiece with several MIMD systems has shown that the overall performance of those systems has been greatly limited by the communication bottlenecks caused by data traffic contention. The research has therefore recognized the need for a hierarchical parallel architecture which can be built in 'modular' form [Morris and Theaker 1987][AL – Sudani 1988]. This technique provides the necessary processing power such that to expand as required with satisfactory upper limit. The promising future engagement of the MIMD architectures to the efficient utilisation of the VLSI technology will furthermore lead to the integration of an intelligent (smart) vision system.

11- Vision is computationally intensive; conventional computers (as mentioned before) are too slow to perform complex visual tasks in real time. As discussed in Section V and in Article 1 of this section, various forms of parallel processing are used to speed up some of the simpler computations commonly performed by vision systems. Yet much less is known about how to speed up the more complex and higher level computations; many of which involve combinatorial searchers. To achieve real time vision in complex environments, it may be important to design architectures and algorithms that can make effective use of paralellism at all levels [Hayat et al. 1992][Morris et al. 1989]. As already pointed out, humans can recognise objects (even complex objects whose presence was unexpected) in a fraction of a second, which is enough time for few hundred cycles of the 'neural hardware' in the human visual system [Rosenfeld 1988]. Current computer vision systems have a long way to go before they will be able to match this performance. However, the T800 transputer introduced by INMOS [INMOS 1986] was hereby found to be attractive single chip computer that facilitates building novel concurrent systems. The associated concurrent language "occam" uses software channels to communicate between transputers. Those channels can be mapped on to the transputer's serial links if each process is to be run on a different transputer. This will ensure for more machine cycles per second essential to speed up the vision system to work in real time.

12- The computer vision system of this research provides a centralised resources management with fault-tolerance. It offers an intertask virtual resources for visual data development. It can be described as flexible, multipurpose, simple, high performance, and cost-effective.

REFERENCES

Aggarwal J.K., and N Nandhakumar (1988), On the Computation of Motion from Sequences of Images - A Review, Proc of the IEEE, Computer Vision, Vol.76, No.8, August.

Agrawal D.P., and Jain R. (1981), A Multiprocessor System for dynamic Scene Analysis, Proc. of Workshop.Comp.Arch.for Pattern Anal.Mach.Intel 1.

Aloimonos J. (1988), Visual Shape Computation, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

AL-Sudani Asaad A. M. (1998i), Image Processing Environment using Parallel Architecture, Proc. of the 4th Iraqi Technological Conference, Univ. of Technology, Baghdad, IRAQ, 11-12 May.

AL-Sudani Asaad A.M. (1998ii), Image Motion Detection Based on Parallelism, Journal of Engineering and Technology, Univ. of Technology, Baghdad, IRAQ, Vol.17, No.3, April.

AL-Sudani Asaad A.M. (1981), Development of Circular Arcs Algorithms using Digital Incremental Stepping on Computer Graphics System, M.Sc. Thesis, Dept. of Elec. Eng., Univ. of Baghdad, Baghdad (IRAQ), November.

AL-Sudani Asaad A.M.(1988), A Bus Extension for a Specific Architecture Network, PhD Transfer Report, Dept. of Computation, UMIST, Manchester (UK), July.

Bajcsy R. (1988), Active Perception , Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Bertero M., Pogglo T.A., and Torre V. (1988), Ill-posed Problem in Early Vision, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Burt J.(1988), Smart Sensing within a Pyramid Vision Machine, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Dew P.M., Earnshaw R.A., and Heywood T.R. (Eds.) (1988), Parallel Processing for Computer Vision and Display, Addison-Wesley Publishing Company,.

Faugeras O.D. (Ed.) (1983), Fundamentals in Computer Vision , Cambridge University Press.

Hayat L., Naqvi A., and Sandier M.B. (1992), A Comparative Evaluation of Fast Thining Algorithms on a Multiprocessor Architecture, Journal of Image and Vision Computing, Vol.10, No.4, pp.210-218.

Hwang K., and Briggs F.A. (1985), Computer Architecture and Parallel processing, McGraw-Hill.

INMOS Group (1986), IMS T800 Transputer, Product Overview, INMOS, November.

Lawton D.T., and McConnell C.C. (1988), Image Understanding Environment, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Lee D. (1988), Some Computational Aspects of Low Level Computer Vision, Proc. of the IEEE, Vol.76, No.8, August.

Li H., and Kender J.R. (1988), Scanning the Issue, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Lin W.M., and Prassana Kumar V.K. (1990), Efficient Programming on SIMD Machines, Computer Vision Graphics and Image Processing, No.49, pp,104-120.

Maresca M., Lavin M.A., and Li H (1988), Parallel Architectures for Vision, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Morris D., and Theaker C.J. (1987), Hierarchical Multiprocessor Architecture, IEE Proceedings, Vol.134, Pt.E, No.4, pp.161-167, July.

Morris D., Theaker C.J., and Phillips R. (1989), A Personal Parallel System-PPS , Technical Report, Dept. of Computation, UMIST, Manchester (UK), January.

Pesl P.J. (1988), Geometric Modeling and Computer Vision, Proc. of the IEEE, Computer Vision, Vol.76, No.8, August.

Rosenfeld A. (1988), Computer Vision: Basic Principles, Proc. Of the IEEE, Computer Vision, Vol.76, No.8, August.

SBC1 Single Board Computer for the VMEbus (1986), Technical Manual – Revision C, VME specialists.

Stout Q.F. (1988), Mapping Vision Algorithms to Parallel Architectures, Proc. of the IEEE, Computer Vision, Vol.76, No., August.

VMEbus (IEEE 1014) (1985), Specification Manual - Revision C, IEEE, February.

Weems C.C. (1988), A Preliminary Report on the performance of the image understanding Architecture on the DARPA Integrated Image Understanding Benchmark, British Computer Society-Parallel Processing Specialist Group, CONPAR 88 Conference, Stream C, pp. 50-57.